書き換えと計算機

Jacques Garrigue

2005 年 8 月 11〜13 日

項書き換えは理論情報科学の比較的に新しい分野 (1970 年代頃から名前が確定された) である。情報科学は論理学を基礎にして来たが、構築される論理体系が複雑になるにつれて、その計算的な意味を理解する必要が生じた。項書き換えで証明されるのは、ある体系の意味が正しいかどうかではなく、その体系の中で計算ができるかどうかである。計算が止らなかったり、計算の仕方によって異る結果が得られたりしたら、そもそもその体系が意味をなさないことになってしまう。

数学ではその現象は定義兼定理の形で出てくる。複雑な定義をすると、その定義にはちゃんと意味があることを証明しなければならない。情報科学は定義を大量に作ってしまうので、その理論が必要になった。

1 書き換え系

項書き換えは数学と違うとは言っても、書き換えは数学のあらゆるところにある。ほとんどの場合、数学の計算は数式か方程式の書き換えという形で行われる。

$$\begin{aligned}\sin\left(\frac{\pi}{3}-\frac{\pi}{4}\right) &= \sin\frac{\pi}{3}\cos\frac{\pi}{4}-\cos\frac{\pi}{3}\sin\frac{\pi}{4}\\ &= \frac{\sqrt{3}}{2}\times\frac{\sqrt{2}}{2}-\frac{1}{2}\times\frac{\sqrt{2}}{2}\\ &= \frac{\sqrt{6}-\sqrt{2}}{4}\end{aligned}$$

上記の方程式では、$=$ は単に同値性を表しているだけではない。段々式が簡単なものにされて行く。書き換えはそういう方向性をもった方程式のことを言う。あるいは、等価式でもよい。

$$(\forall x>0)\qquad x^2-1=0\Leftrightarrow x^2=1\Leftrightarrow x=1$$

書き換えとして書くのであれば、初めの例で=を、二番目の例で $\Leftrightarrow$ を、書き換えを表す $\rightarrow$ に置き換える。

書き換え系とは、規則によって可能な書き換えを完全に表したものを言う。簡単な例として、$\mathbf{Z}_{/3\mathbf{Z}}$ の足し算を見よう。

$$\begin{array}{lcllcllcl} 0+0 & \rightarrow & 0 & 0+1 & \rightarrow & 1 & 0+2 & \rightarrow & 2\\ 1+0 & \rightarrow & 1 & 1+1 & \rightarrow & 2 & 1+2 & \rightarrow & 0\\ 2+0 & \rightarrow & 2 & 2+1 & \rightarrow & 0 & 2+2 & \rightarrow & 1 \end{array}$$

この規則は式のどの部分に対しても適用できる。E' は E の一箇所を上記の規則で書き換えた式であれば、また $E\rightarrow E'$ と書く。

書き換えの例

$$\begin{aligned}&(1+2)+(1+(1+0))\\ &\rightarrow 0+(1+(1+0))\rightarrow 0+(1+1)\rightarrow 0+2\rightarrow 2\end{aligned}$$

ちなみに、上の規則では、$0+(1+(1+0)) \to 1+(1+0)$ とは書けない。書き換えは見えるものの話であって、その意味の話ではない。

こういう書き換え系を定義したとき、その定義が使えるかどうかどうかを考えなければならない。そのために二つの性質が望ましい。

まず、書き換えを繰り返したときそれが無限に続くことはないという**停止性**。それと、書き換えをどの順番で行っても結果が変わらないという**合流性**。

書き換える式の中の+記号を数えれば、上の規則が必ず止るということはすぐ分かる。毎回記号の数が減っていくので、元々あった記号の数の分の回数しか計算が行えない。式の中の何かを数えるのは、停止性を証明する最も一般的な方法である。(しかし何を数えればいいか分からないことが多い)

合流性はもう少し複雑になっている。最初の段階では、1+2 ではなく 1+0 を書き換えてもよかった。そうすると、別の書き換えの列ができる。その支流がやがて合流することを証明しなければならない。

式だけを見て証明する方法と、式の意味を考えて証明する方法がある。ここでは、式の意味が明かであるから、それを使う。

$$S(E) = \{E \text{ の中の数字の } \mathbf{Z}_{/3\mathbf{Z}} \text{における総和 }\}$$

とすると、$E \to E'$ のとき、$S(E) = S(E')$ が成り立つ。さらに、式は 0,1,2 と+でしか作られていないので、式が一つの数字でない限り書き換えが続けられる。停止性があるので、任意の式 E に対して、$E \to E_1 \to \ldots \to E_n$ の書き換えの列が存在し、$S(E_n) = E_n$ となっている。しかも、上の事から、$S(E) = S(E_n)$。よって、任意の式 E は有限回の書き換えで $S(E)$ に変えられる。しかもそれは書き換えの順序に影響されない。

もう気が付いたと思うが、ここで証明したことは合流性より強い。それは停止性と合流性が同時に成り立つときの性質で、**完備性**という。完備性が成り立つと、任意の式 E に対して、$E{\downarrow}$ と書く**正規形**が存在する。

2 抽象書き換え系

さて、前節で出てきた概念を整理しよう。書き換え系の性質を定義するのに、それをさらに簡単にした**抽象書き換え系**を使う。

定義 1 抽象書き換え系とはある集合 A 上の二個関係 R である。

a と b が A の要素で $R(a,b)$ が成り立つことを $a \to_R b$ と書く。その推移閉包 $(R(a_0,a_1),$ $\ldots, R(a_{n-1},a_n), n>0)$ を $a_0 \xrightarrow{+}_R a_n$ と書く。反射閉包 ($R(a,b)$ または $a=b$) を $a \xrightarrow{=}_R b$ と書く。両方の閉包を併わせた関係 (反射推移閉包,$R(a_0,a_1),\ldots,R(a_{n-1},a_n), n \geq 0$) を $a_0 \xrightarrow{*}_R a_n$ と書く。書き換え系の意味はその反射推移閉包によって与えられる。

定義 2 (停止性) A の全ての要素 a に対して、$a \to_R a_1 \to_R \ldots$ となるような無限の書き換えの列が存在しなければ、R は停止性をもつと言う。

定義 3 (正規形) A の要素 a に対して、$a \to_R b$ なるような b が存在しなければ、a は R の正規形だと言う。

定義 4 (合流性)

1. A の任意の要素 a,b,c について $a \xrightarrow{*}_R b$ と $a \xrightarrow{*}_R c$ が成り立つと、必ずある A の要素 d が存在し、$b \xrightarrow{*}_R d$ と $c \xrightarrow{*}_R d$ が成り立てば、R は**合流性**をもつと言う。
2. $\forall a,b,c \in A,\ (a \to_R b \wedge a \to_R c) \Rightarrow (\exists d \in A,\ b \xrightarrow{=}_R d \wedge c \xrightarrow{=}_R d)$ ならば、R は**強合流性**をもつと言う。
3. $\forall a,b,c \in A,\ (a \to_R b \wedge a \to_R c) \Rightarrow (\exists d \in A,\ b \xrightarrow{*}_R d \wedge c \xrightarrow{*}_R d)$ ならば、R は**弱合流性**をもつと言う。

$\rightarrow_R$ からその対象閉包 $\leftrightarrow_R$ を作る。

定義 5 (CR 性) $\forall a, b \in A,\ a \overset{*}{\leftrightarrow}_R b \Rightarrow (\exists c \in A,\ a \overset{*}{\rightarrow}_R c \wedge b \overset{*}{\rightarrow}_R c)$ ならば、R はチャーチ・ロッサー性 *(CR 性)* をもつと言う。

定理 1 (合流性の性質)
1. *CR* 性と合流性は等価である。
2. R は強合流すれば、R は合流する。
3. R は停止し、かつ弱合流すれば、R は合流する。*(Newman の補題)*

定義 6 (完備性) R は停止性と合流性を両方もっていれば、R は完備だと言う。

R は完備な書き換え系だと、全ての要素 a に対して一つだけの正規形が存在し、それを $a{\downarrow_R}$ と書く。

3 項書き換え系

上の $\mathbf{Z}_{/3\mathbf{Z}}$ の例では有限個の数字しか扱えなかった。書き換えは式のどの部分にも適用できるので、式はいくら大きくても構わないが、こういう弱い形の規則では複雑なことはできない。

項書き換えでは、規則の中で変数も使える。それによって、一つな規則で無限な書き換えができる。

まず、自然数の足し算を行う書き換え系を定義しよう。

自然数は無限なので、直接表現できない (無限個の書き換え規則が必要になる)。Peano の表現法を使う。

$$\overline{n} = \begin{cases} z & n = 0 \\ s(\overline{n-1}) & n > 0 \end{cases}$$

例えば、$\overline{3} = s(s(s(z)))$ になる。

足し算は次の二つの規則で定義できる。

$$\begin{aligned} z + X &\rightarrow X \\ s(X) + Y &\rightarrow X + s(Y) \end{aligned}$$

例

$$s(s(z)) + s(z) \rightarrow s(z) + s(s(z)) \rightarrow z + s(s(s(z))) \rightarrow s(s(s(z)))$$

上を見れば項書き換えで扱っている**項**というのは、数学の**式**のようなものである。ただ、形式化のために、それを厳格に定義する必要がある。

定義 7 (項) $\Sigma = \mathcal{F} \cup \mathcal{V}$ の記号の集合の上に項の集合を定義する。

1. $\mathcal{V}$ は加算無限個の変数の集合である。
2. $\mathcal{F}$ は有限個の関数記号の集合である。各関数記号 f は定められた項数をもっており、項数 n の関数記号の集合を $\mathcal{F}_n$ と書く。$m \neq n$ ならば、$\mathcal{F}_m \cap \mathcal{F}_n = \emptyset$。
3. Σ 上の項の集合 $\mathcal{T}(\Sigma)$ は以下の条件を満たす最小の集合である。
   (a) $\mathcal{V} \subseteq \mathcal{T}(\Sigma)$
   (b) $f \in \mathcal{F}_n$ および $t_1, \ldots, t_n \in \mathcal{T}(\Sigma)$ ならば $f(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma)$

t は $\mathcal{T}(\Sigma)$ の項ならば、$\mathcal{V}(t)$ は t の中に表われる変数の集合である。$\mathcal{V}(t)$ は空集合ならば、t を**基礎項**と呼ぶ。

定義 8 (項書き換え系) 記号の集合 Σ と書き換え規則の集合 R の対 (Σ, R) を項書き換え系と言う。$R = \{s_i \to t_i \mid s, t \in \mathcal{T}(\Sigma), i \in I\}$ は項の対 (s_i, t_i) から作る書き換え規則 $s_i \to t_i$ の集合で、$s_i, t_i, i \in I$ は次の条件を満す。

1. $s_i \in \mathcal{V}$
2. $\mathcal{V}(t_i) \subseteq \mathcal{V}(s_i)$

定義 9 (代入) $\mathcal{T}(\Sigma)$ 上の代入は $\mathcal{V}$ から $\mathcal{T}(\Sigma)$ への任意の有限な部分関数 σ である。次のように σ は $\mathcal{T}(\Sigma)$ から $\mathcal{T}(\Sigma)$ への写像 $\overline{\sigma}$ に拡張される。

$$\overline{\sigma}(t) = \begin{cases} t & t \in \mathcal{V}, t \notin \mathcal{D}_\sigma \\ t' & t = X \in \mathcal{V}, \sigma(X) = t' \\ f(t'_1, \ldots, t'_n) & t = f(t_1, \ldots, t_n), \overline{\sigma}(t_i) = t'_i \end{cases}$$

σ と $\overline{\sigma}$ は同一視される。

文脈とは、ある一つの出現位置が空になっている項である。_ で空の項を表すと、例えば、$s(s(\_)+X)+s(Y)$ は文脈である。任意の文脈を $C[\,]$ で表す。

$\mathcal{T}(\Sigma)$ 上の任意の二項関係 R は

$$\forall s, t \in \mathcal{T}(\Sigma)((s,t) \in R \Rightarrow (C[s], C[t]) \in R)$$

のとき、**両立性**があると言う。

書き換え規則 $r : s \to t$ 次の $\mathcal{T}(\Sigma)$ 上の二項関係 $\to_r$ を定義する。$\to_r$ は関係 $\{(\sigma(s), \sigma(t)) \mid \sigma$は $\mathcal{T}(\Sigma)$ 上の任意の代入 $\}$ を含み、両立性を満す最小の二項関係である。いいかえると、書き換え規則 $r : s \to t$ は

$$\text{集合} \to_r = \{(C[\sigma(s)], C[\sigma(t)]) \mid \text{代入}\sigma, \text{文脈}\, C[\,]\}$$

を定義している。

$\to_R = \bigcup_{r \in R} \to_r$ は項書き換え系 $(\mathcal{T}(\Sigma), R)$ が作る書き換え関係であり、$(\mathcal{T}(\Sigma), \to_r)$ は抽象書き換え系である。

4 正則項書き換え系

一般的にはある項書き換え系が完備であるかどうかを判断する方法はない。しかし、合流性または停止性について十分な条件はいくつか知られている。合流性に関して正則性はその一つである。

正則性は線型性と重なりのなさから定義される。

ある項の中で、同じ変数が二回以上現われていなければ、その項は**線型**だと言う。同様に、ある書き換え規則の左辺 (右辺) が線型な項であれば、その書き換え規則は左線型 (右線型) だと言う。ある項書き換え系の全ての規則が左線型であれば、その系も左線型と言われる。

重なりとは、項を木で表現したときの木同士の重なりである。それを性格に定義するために、出現位置を先に定義しなければならない。

$\mathbf{N}_+^*$ を正の自然数の有限列の集合とする。長さ 0 の列の集合は $N_+^0 = \{\epsilon\}$、長さ k の列の集合が $\mathbf{N}_+^k = \{n_1 n_2 \ldots n_k \mid n_1, n_2, \ldots, n_k \in \mathbf{N}_+\}$ なので、$\mathbf{N}_+^* = \bigcup_{k \geq 0} \mathbf{N}_+^k$。$u = u_1 \ldots u_k \in \mathbf{N}_+^k$ と $v = v_1 \ldots v_l \in \mathbf{N}_+^l$ があれば $u \cdot v = u_1 \ldots u_k v_1 \ldots v_l \in \mathbf{N}_+^{k+l}$ は u と v の連結という。

任意の項 t に対して、出現位置の集合 $\mathcal{O}(t) \subset \mathbf{N}_+^*$ と出現位置 $u \in \mathcal{O}(t)$ における部分項 t/u を次のように定義する。

1. $t = X \in \mathcal{V}$ のとき、$\mathcal{O}(t) = \{\epsilon\}$、$t/\epsilon = t$
2. $t = f(t_1, \ldots, t_n)$ のとき、
   $\mathcal{O}(t) = \{\epsilon\} \cup \{i \cdot u \mid i \leq n, u \in \mathcal{O}(t_i)\}$、$t/\epsilon = t$、$t/i \cdot u = t_i/u$

出現位置を使えば、重なりを以下のように定義できる。二つの書き換え規則 $r_1 : s_1 \to t_1$, $r_2 : s_2 \to t_2$ について、r_2 は次の条件を満すとき、r_1 に重なると言う。

$$\exists u \in \mathcal{O}(s_1), \exists \text{ 代入} \sigma, \sigma(s_1/u) \notin \mathcal{V} \text{ かつ} \sigma(s_1/u) = \sigma(s_2)$$

ただし、r_1 と r_2 は同一の書き換え規則のとき、$u \neq \epsilon$ とする。

この場合、s_1/u と s_2 は**単一化可能**であり、σ を単一化代入と言う。

定義 10 左線型かつ重なりのない項書き換え系を正則項書き換え系と言う。

正則な例

$$\begin{aligned} R &= \{f(0) \to 1, f(s(X)) \to g(s(X), f(X))\} \\ \Sigma &= \{f, g, 0, 1, X\}, X \in \mathcal{V} \end{aligned}$$

重なりをもった例

$$R = \{f(f(X)) \to g(X), f(g(X)) \to h(X)\}$$

定理 2 正則書き換え系は合流性をもつ。

5 危険対

正則書き換え系において、書き換え規則に重なりがないと仮定した。しかし、実際には項書き換え系が与えられたとき、重なりがあることが多い。そういう場合でも、合流性の十分条件を見つけなければならない。

条件を表現するのに、二つの規則に重なりがあるとき生じる**危険対**という概念が使われる。

定義 11 $r_1 : s_1 \to t_1$, $r_2 : s_2 \to t_2$ を書き換え規則とする。r_2 が出現位置 u で *(*最も一般的な*)* 代入 σ により、r_1 と重なるとする。このとき得られる項の対

$$(\sigma(s_1)[u \leftarrow t_2], \sigma(t_1))$$

を r_1 と r_2 の危険対と言う。

定理 3 (Σ, R) を項書き換え系、CP を R の危険対の集合とする。

$$(\Sigma, R) \text{ は弱合流性をもつ} \Leftrightarrow \forall (p, q) \in CP, \exists s, (p \xrightarrow{*} s \wedge q \xrightarrow{*} s)$$

定理 4 (Σ, R) は停止性をもつ項書き換え系、CP を R の危険対の集合とする。

$$(\Sigma, R) \text{ は合流性をもつ} \Leftrightarrow \forall (p, q) \in CP, \widehat{p} = \widehat{q}$$

ここで $\widehat{p}$ を p のある正規形とする。

```
fun analyze ≻ s t =
  if s ≻ t then
    (s,t)
  else if t ≻ s then
    (t,s)
  else raise "完備化は失敗した"

fun complete ≻ E =
  begin
    var R = {analyze ≻  s t  |  s = t ∈  E}
    var CP = R の危険対の集合
    while CP ≠ ∅ do
      begin
        val (c,p) = CP の任意の対;
        CP ← CP \ {(c,p)};
        val p̂ = p の R に関しての正規形
        val q̂ = q の R に関しての正規形;
        if p̂  =  q̂ then
          begin
            val (s,t) = analyze ≻ p̂ q̂;
            R ← R ∪ {(s,t)};
            CP ← CP  ∪  (⋃r ∈ R)  (s → t と r の危険対の集合)
          end
      end
    R
  end
```

図 1: Knuth-Bendix の完備化手続き

この定理は停止性をもつ項書き換え系の合流性を証明する方法を提供している。(Σ, R) は停止性をもつとする。

1. R の危険対を全て求める。
2. 危険対 (p, q) について、p, q の正規形 $\widehat{p}, \widehat{q}$ を求める。停止性があるのでできる。
3. $\widehat{p} = \widehat{q}$ であるかどうかを調べる。
4. 全ての (p, q) について、$\widehat{p} = \widehat{q}$ ならば、(Σ, R) は合流性をもつ。

6 完備化手続き

項書き換え系において完備性は重要な性質である。しかし、ある問題に対して、直接にそのための項書き換え系を書いても、それが完備でないことが多い。あるいは、元より問題は項書き換え系の形ではなく、等式の形で書かれることもある。

そういう問題から完備な項書き換え系を作る完備化手続きが存在する。当然、ある項書き換え系が完備かどうかを自動的に判断するのは不可能なので、その手続きは全ての場合で成功するわけではない。しかし、多くの場合では、期待どおりの結果が得られる。

完備化手続きは、ある簡約順序 (項の上の厳格半順序) と有限な等式集合をもって開始し、危険対を生成しては必要な方向付けを行うことにより、等式集合が表す理論に対する有限正則書き換え系を求めようとする。危険対 $s = t$ の両辺を簡約して等式 $u = v$ を得たとき、u と v が相異なるならば、書き換えによる証明を $s = t$ に与えるために、新しい規則 $u \to v$ か $v \to u$ を生成する。規則の方向を決定するためには、与えられた簡約順序が使われる。すなわち、もし $u \succ v$ ならば $u \to v$ を追加し、$v \succ u$ ならば $v \to u$ を追加する。この新しい規則を用いて新しい危険対を生成する。手続きを実行すると、次のような三つの場合がある。

1. 正則書き換え系を発見することに成功する。
2. 失敗する。
3. 無限個の規則を生成し続ける。

完備化手続きを簡単なプログラミング言語で書いたものを図 1 で示した。

この手続きは人間が実行するのがほとんど不可能である。しかし、計算機では効率よく実行できる。以下に群論の公理を項書き換え系に変える例を載せた。

群論の公理

$$\begin{aligned} 0 + a &= a \\ (-a) + a &= 0 \\ (a + b) + c &= a + (b + c) \end{aligned}$$

より次の正則書き換え系が得られる

$$\begin{aligned} 0 + a &\to a & a + (-a) &\to 0 \\ (-a) + a &\to 0 & a + ((-a) + b) &\to b \\ (a + b) + c &\to a + (b + c) & -0 &\to 0 \\ (-a) + (a + b) &\to b & -(-a) &\to a \\ a + 0 &\to a & -(b + a) &\to (-a) + (-b) \end{aligned}$$

参考文献

[1] Nachum Dershowitz and Jean-Pierre Jouannaud. Chapter 6, rewriting systems. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*. Elsevier Science, 1990.

[2] Gérard Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, Vol. 27, No. 4, pp. 797–821, October 1980.

[3] 井田哲雄. 計算モデルの基礎理論, 岩波講座ソフトウェア科学, 第 12 巻, 6 章, 書換えモデル, pp. 223–296. 岩波書店, 1991.

```
1 : 0+a = a
2 : (-a)+a = 0
3 : (a+b)+c = a+(b+c)
4 : (-a)+(a+b) = b
5 : (-0)+a = a
6 : (-(-a))+0 = a
7 : (-(a+b))+(a+(b+c)) = c
8 : (-(-a))+b = a+b
Rule 6 deleted
9 : a+0 = a
10 : -((-(a+b))+a) = b
11 : (-(a+(-b)))+a = b
12 : (-(a+(b+c)))+(a+(b+(c+d))) = d
13 : (-(a+(-b)))+(a+c) = b+c
14 : a+(-a) = 0
15 : a+((-a)+b) = b
16 : -0 = 0
Rule 5 deleted
17 : -(-a) = a
Rule 8 deleted
18 : a+((-(b+a))+b) = 0
19 : -((-(a+(b+c)))+(a+b)) = c
20 : a+((-(b+a))+(b+c)) = c
21 : -((-b)+(-a)) = a+b
22 : -(a+(-(b+a))) = b
23 : (-(a+(b+(-c))))+(a+b) = c
24 : a+(-((-b)+a)) = b
25 : (-(b+((-(a+c))+a)))+b = c
26 : (-(a+(b+(c+d))))+(a+(b+(c+(d+e)))) = e
27 : (-(a+(b+(-c))))+(a+(b+d)) = c+d
28 : (-(b+((-(a+c))+a)))+(b+d) = c+d
29 : a+((-(b+(c+a)))+(b+(c+d))) = d
30 : a+((-((-b)+a))+c) = b+c
31 : a+(b+(-(a+b))) = 0
32 : (-(a+b))+a = -b
Rule 28 deleted
Rule 25 deleted
Rule 18 deleted
Rule 11 deleted
Rule 10 deleted
33 : (-(a+(b+c)))+(a+b) = -c
Rule 23 deleted
Rule 19 deleted
34 : -(b+(-a)) = a+(-b)
Rule 22 deleted
Rule 21 deleted
Rule 13 deleted
35 : a+(b+((-(a+b))+c)) = c
36 : (-(a+b))+(a+c) = (-b)+c
Rule 33 deleted
Rule 29 deleted
Rule 27 deleted
Rule 26 deleted
Rule 20 deleted
Rule 12 deleted
Rule 7 deleted
37 : a+(b+(-((-c)+(a+b)))) = c
38 : -((-b)+a) = (-a)+b
Rule 37 deleted
Rule 30 deleted
Rule 24 deleted
39 : a+(b+(c+(-(a+(b+c))))) = 0
40 : a+(-(b+a)) = -b
Rule 31 deleted
41 : -(b+a) = (-a)+(-b)
Rule 40 deleted
Rule 39 deleted
Rule 38 deleted
Rule 36 deleted
Rule 35 deleted
Rule 34 deleted
Rule 32 deleted
Canonical set found :
1 : 0+a = a
2 : (-a)+a = 0
3 : (a+b)+c = a+(b+c)
4 : (-a)+(a+b) = b
9 : a+0 = a
14 : a+(-a) = 0
15 : a+((-a)+b) = b
16 : -0 = 0
17 : -(-a) = a
41 : -(b+a) = (-a)+(-b)
```

図 2: 完備化手続の実行履歴